MACHINE LEARNING – 2
BUSINESS REPORT

THANUSRI A

11-02-2024

PROBLEM 1

Define the problem and perform Exploratory Data Analysis

Problem definition - Check shape, Data types, and statistical summary - Univariate analysis - Multivariate analysis - Use appropriate visualizations to identify the patterns and insights - Key meaningful observations on individual variables and the relationship between variables

Data Pre-processing

Prepare the data for modelling: - Outlier Detection(treat, if needed)) - Encode the data - Data split - Scale the data (and state your reasons for scaling the features)

Model Building

Metrics of Choice (Justify the evaluation metrics) - Model Building (KNN, Naive bayes, Bagging, Boosting)

Model Performance evaluation

Check the confusion matrix and classification metrics for all the models (for both train and test dataset) - ROC-AUC score and plot the curve - Comment on all the model performance

Model Performance improvement

Improve the model performance of bagging and boosting models by tuning the model - Comment on the model performance improvement on training and test data

Final Model Selection

Compare all the model built so far - Select the final model with the proper justification - Check the most important features in the final model and draw inferences.

Actionable Insights & Recommendations

Compare all four models - Conclude with the key takeaways for the business

The top five rows of the data set:

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male

The last five rows of the data set:

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	1521	Conservative	67	5	3	2	4	11	3	male
1521	1522	Conservative	73	2	2	4	4	8	2	male
1522	1523	Labour	37	3	3	5	4	2	2	male
1523	1524	Conservative	61	3	3	1	4	11	2	male
1524	1525	Conservative	74	2	3	2	4	11	0	female

The shape of the data set:

```
(1525, 10)
```

There are 1525 rows and 10 columns

Baisc info about the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              1525 non-null   int64
 1   vote                    1525 non-null   object
 2   age                     1525 non-null   int64
 3   economic.cond.national  1525 non-null   int64
 4   economic.cond.household 1525 non-null   int64
 5   Blair                   1525 non-null   int64
 6   Hague                   1525 non-null   int64
 7   Europe                  1525 non-null   int64
 8   political.knowledge     1525 non-null   int64
 9   gender                  1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

- All features are of integer type except for ‘vote’ and ‘gender ‘
- There is no use of the column “Unamed :0” in the data set since it just represents the Id of the person, so we remove the feature “ Unamed:0”

- There are no null values present in the dataset.

```
vote                    0
age                     0
economic.cond.national  0
economic.cond.household 0
Blair                   0
Hague                   0
Europe                  0
political.knowledge     0
gender                  0
dtype: int64
```

- There are 8 duplicate rows present in the data set. So we remove all the duplicate rows present in the data set.

Data summary :

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1517	2	Labour	1057	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1517.0	NaN	NaN	NaN	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	NaN	NaN	NaN	3.245221	0.881792	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	NaN	NaN	NaN	3.137772	0.931069	1.0	3.0	3.0	4.0	5.0
Blair	1517.0	NaN	NaN	NaN	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	NaN	NaN	NaN	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	NaN	NaN	NaN	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	NaN	NaN	NaN	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0
gender	1517	2	female	808	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Now we separate the numerical and categorical features seperately to understand better

```
['vote', 'gender']
['age', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge']
```

- In here, ‘vote’ and ‘gender’ are the categorical features and the rest of them are numerical features.

- Now we check whether we have any ‘?’ in the data values present in any of the features present.
- There are no such values in the data set.

Data summary of the numerical features:

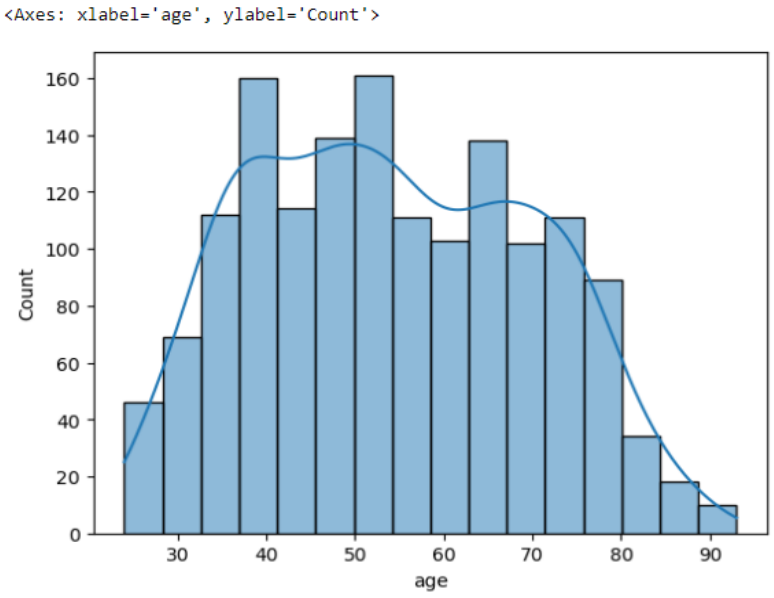
	count	mean	std	min	25%	50%	75%	max
age	1517.0	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	3.245221	0.881792	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	3.137772	0.931069	1.0	3.0	3.0	4.0	5.0
Blair	1517.0	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0

Data summary of the categorical features :

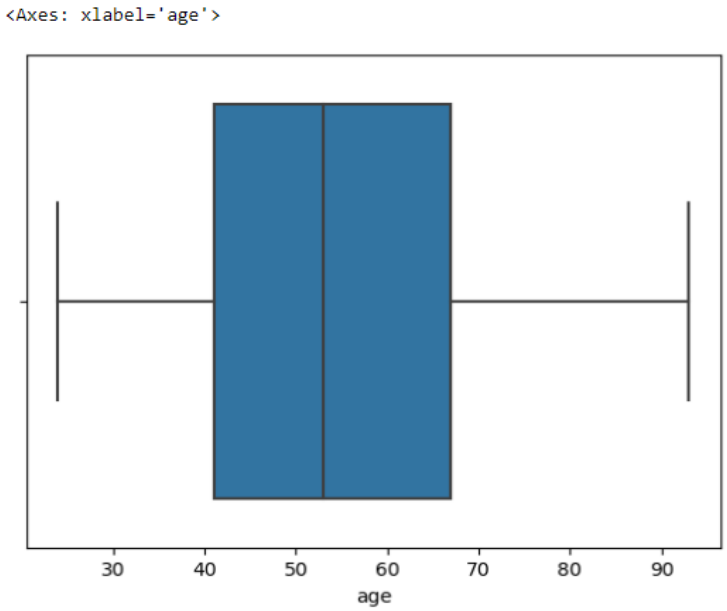
	count	unique	top	freq
vote	1517	2	Labour	1057
gender	1517	2	female	808

UNIVARIATE ANALYSIS :

The histogram for ‘age’ :

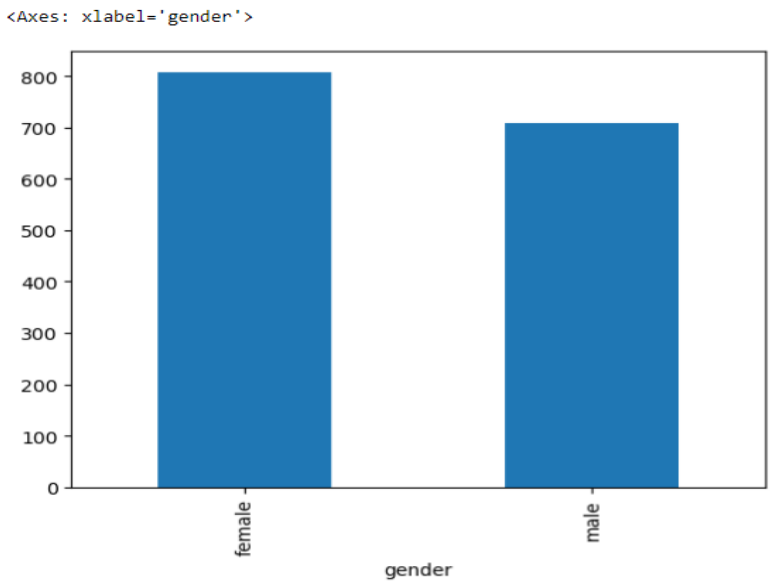


The boxplot for ‘age’



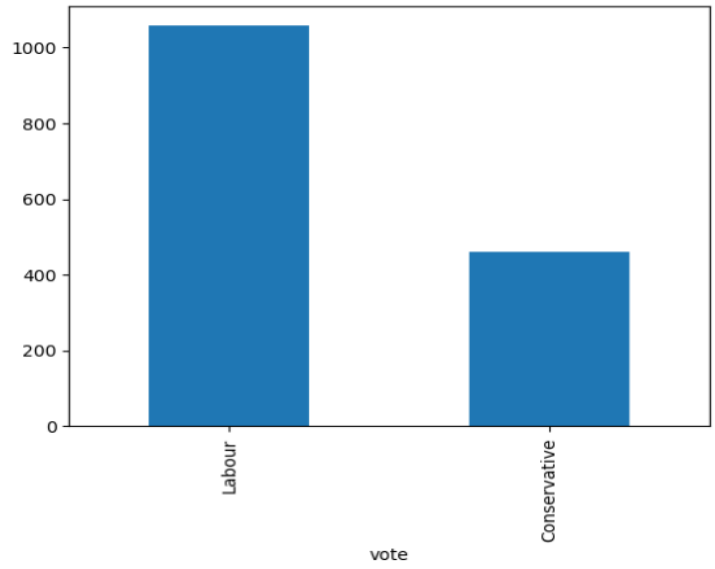
- ‘age’ is the only integer variable and it is not having **outliers.**Also, the dist. plot shows thatthe variable is **normally distributed.**

Frequency distribution of the categorical variables :



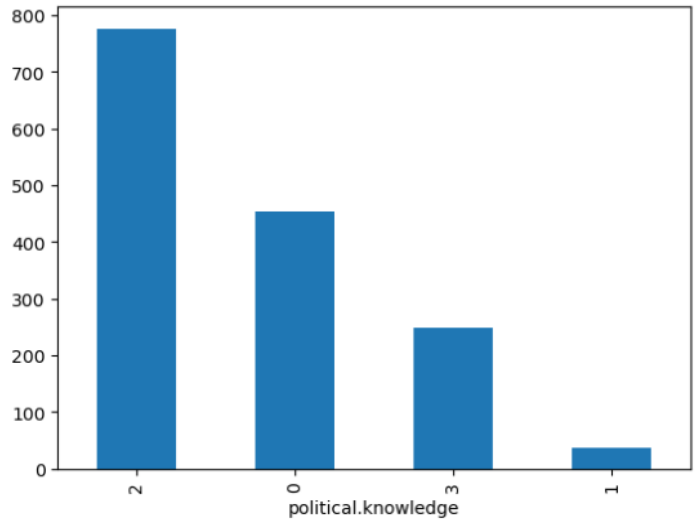
From the above plot we observe that the number of female voters are more than compared to male.

```
<Axes: xlabel='vote'>
```

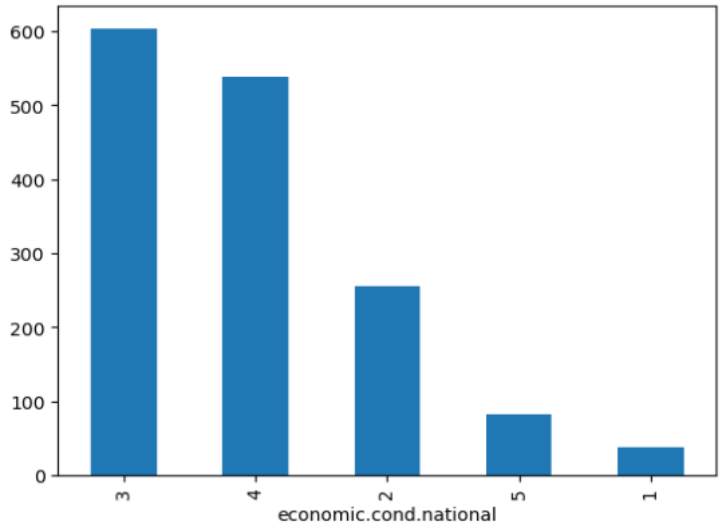


From the above plot, votes are large in number for labour

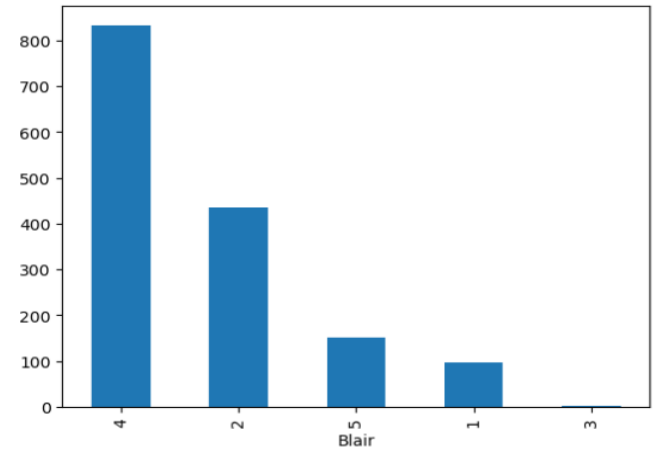
```
<Axes: xlabel='political.knowledge'>
```



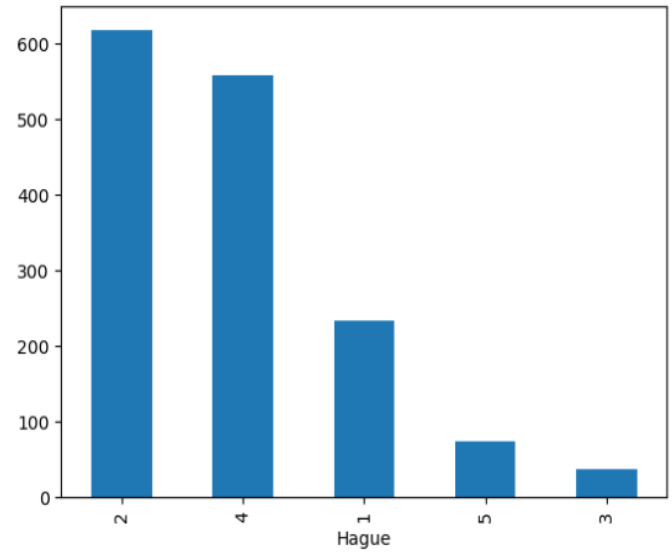

```
<Axes: xlabel='economic.cond.national'>
```



<Axes: xlabel='Blair'>

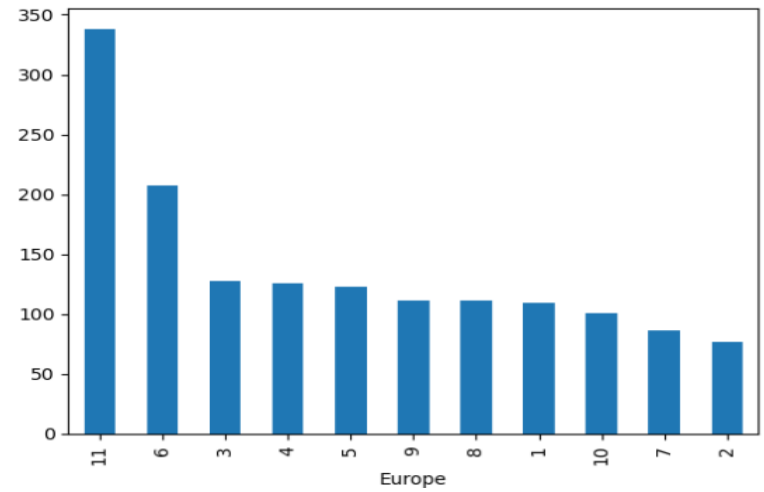


<Axes: xlabel='Hague'>

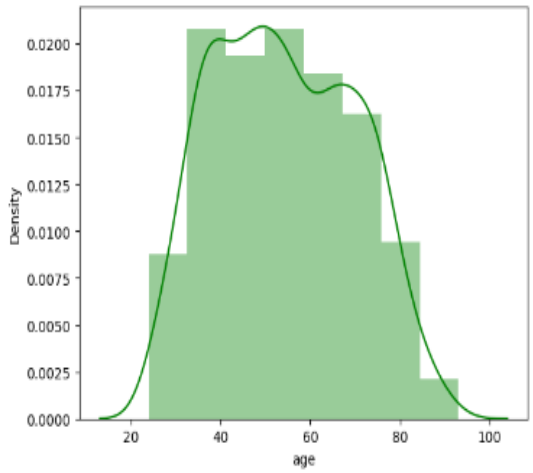


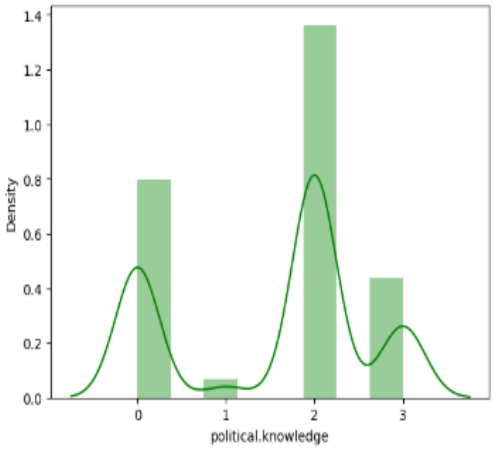
- As the frequency distribution suggests most of the people gave 4 stars to‘Blair’and there are largernumber of people gave 2 stars to ‘Hague’ which made an impact in the dependent variable ‘vote’.

<Axes: xlabel='Europe'>

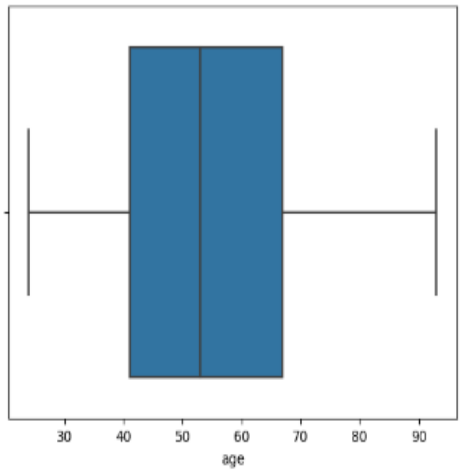


The histogram with kernel density function plot and the boxplot for various features is as follows :

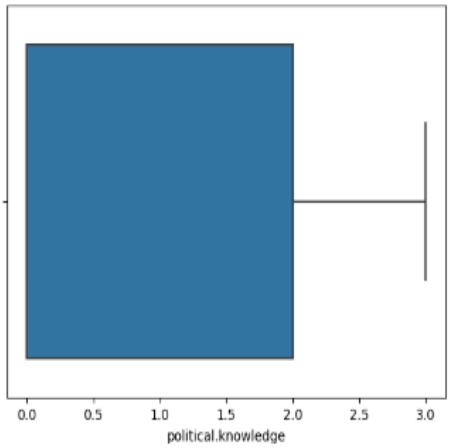


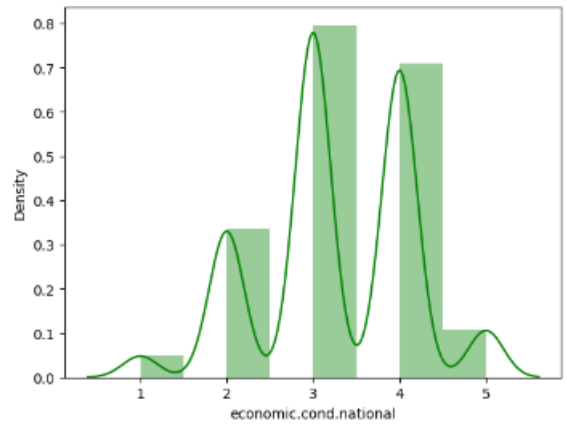


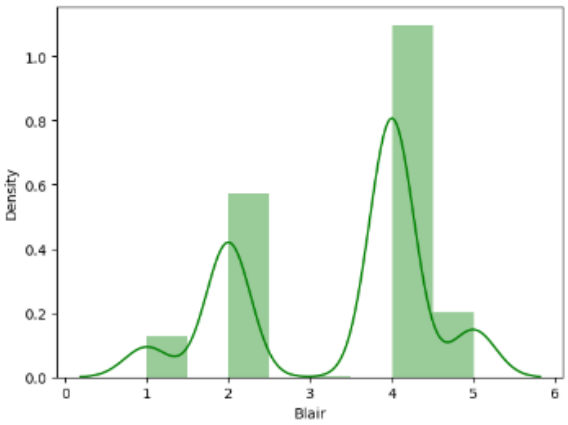
BoxPlot of age



BoxPlot of political.knowledge

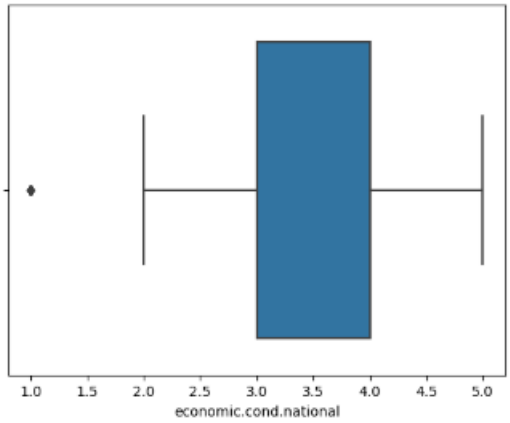


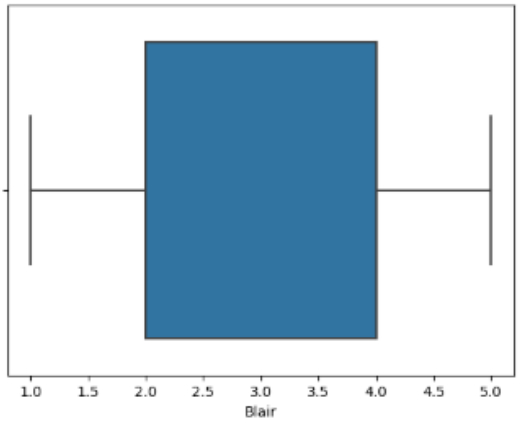


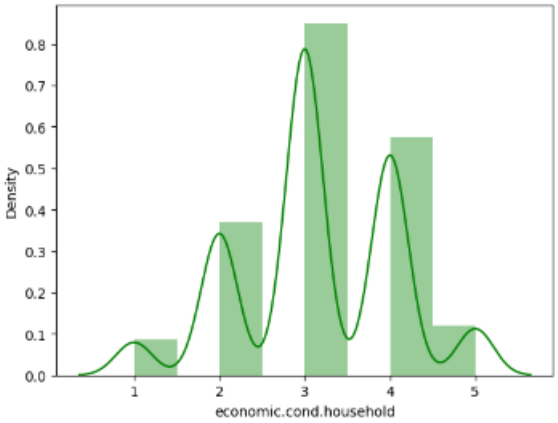


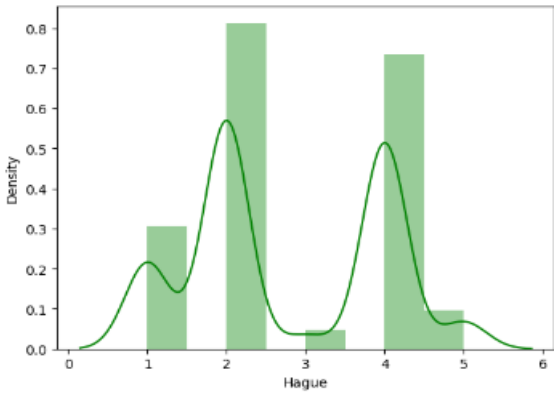
BoxPlot of economic.cond.national

BoxPlot of Blair



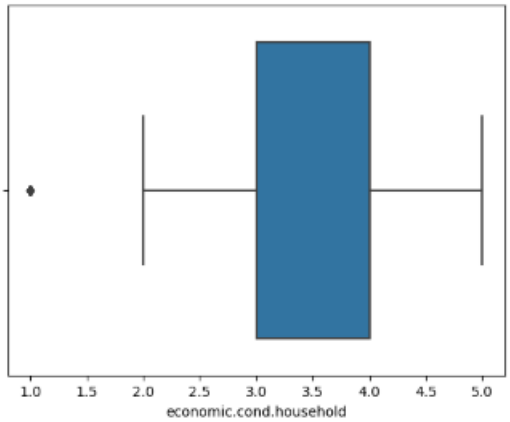


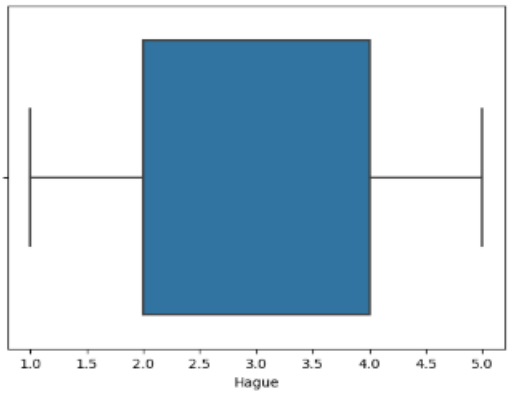


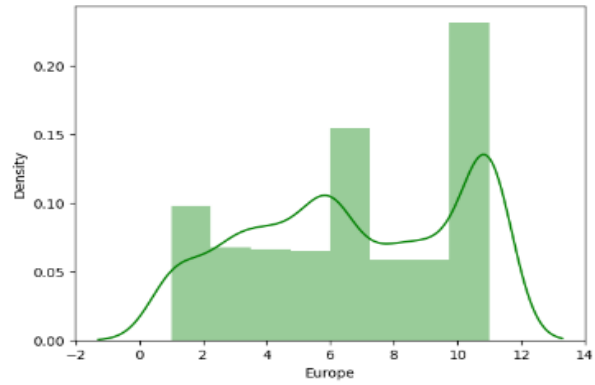


BoxPlot of economic.cond.household

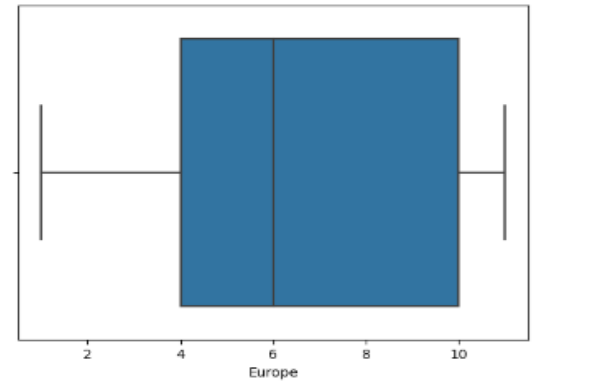
BoxPlot of Hague



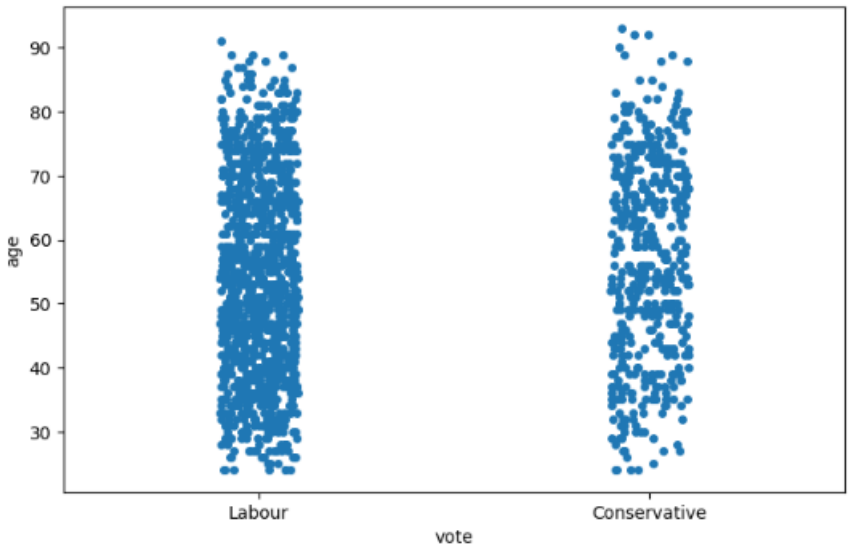


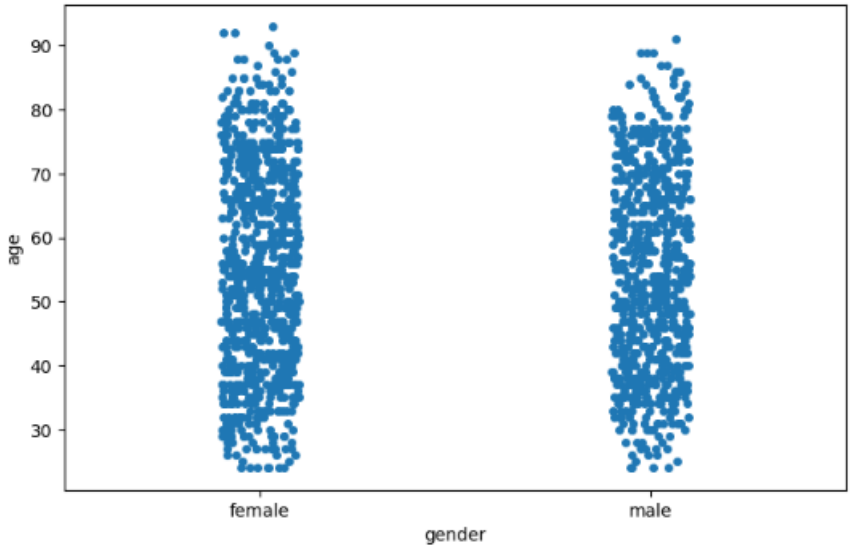


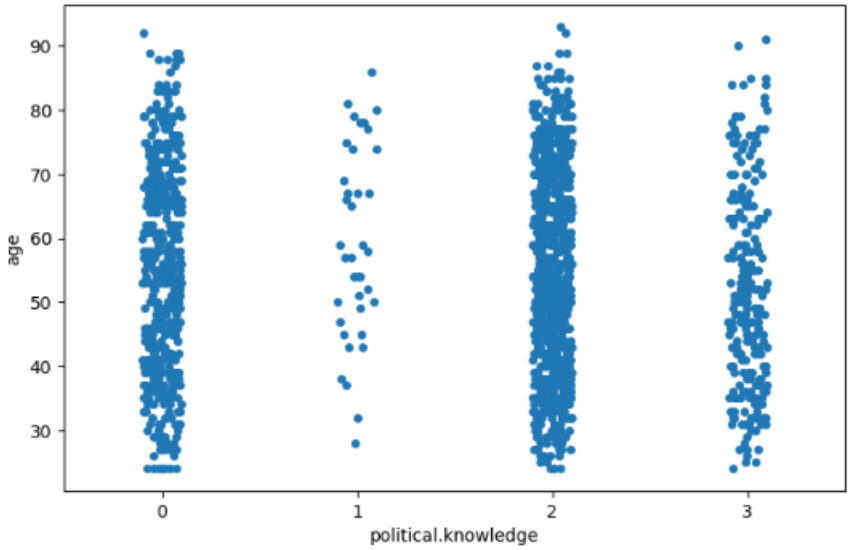
BoxPlot of Europe



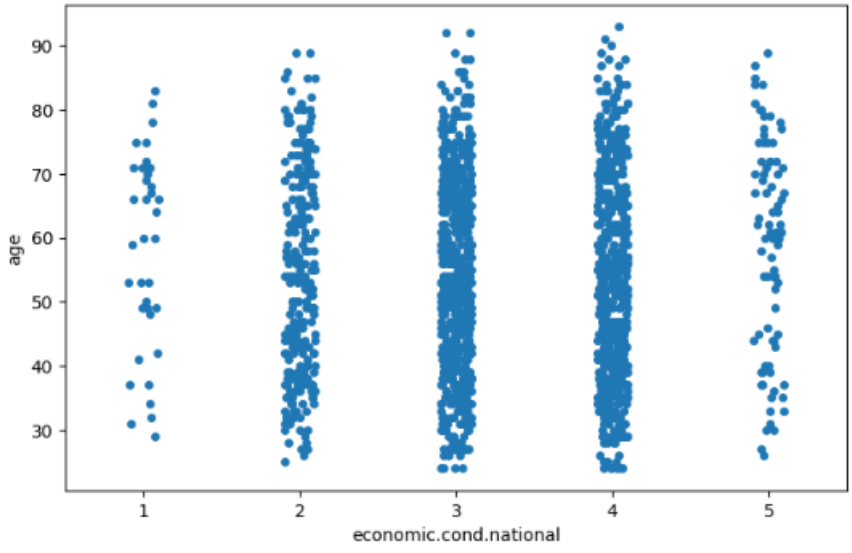
Stripplots between the different features is as follows :

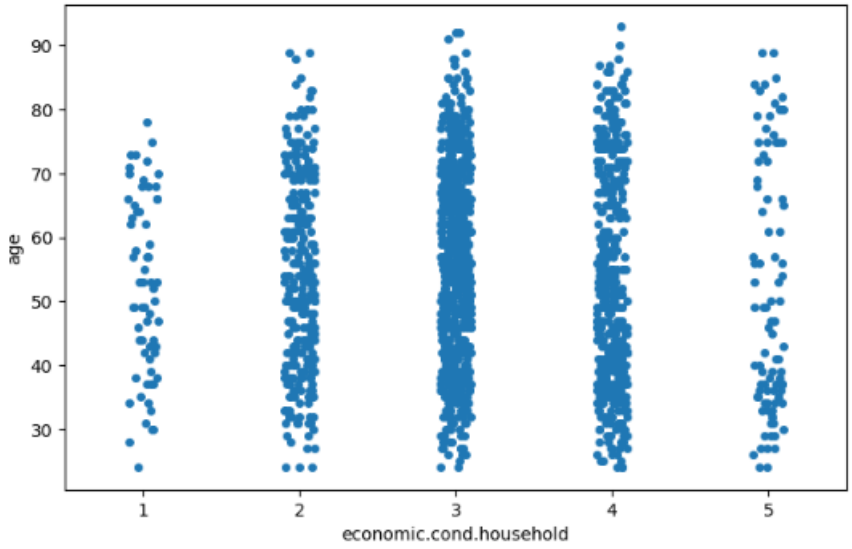






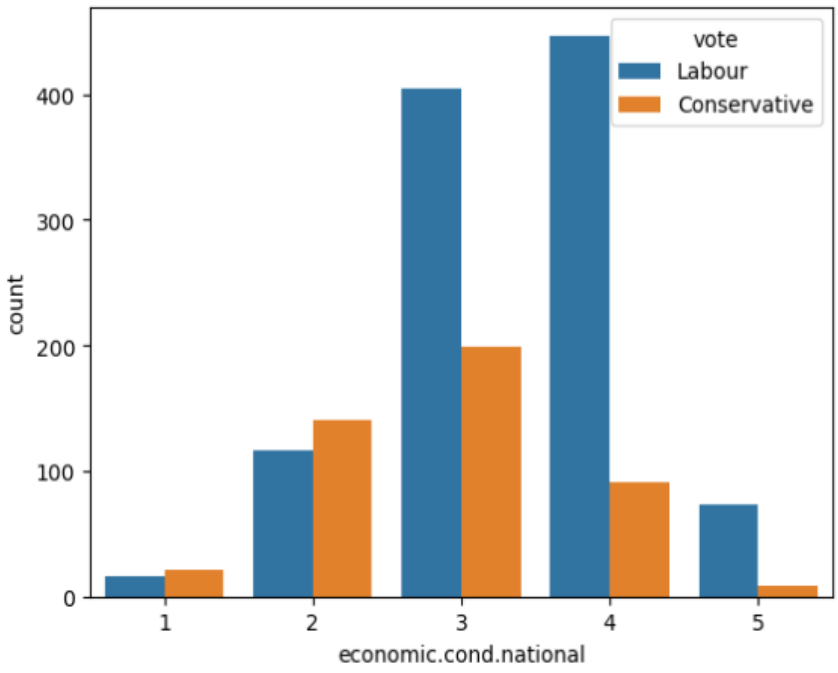
Here, we can see that majority of the population has a moderate understanding of the political situation . however. The middel- aged (35-50) population seem to have a better understanding than the others.



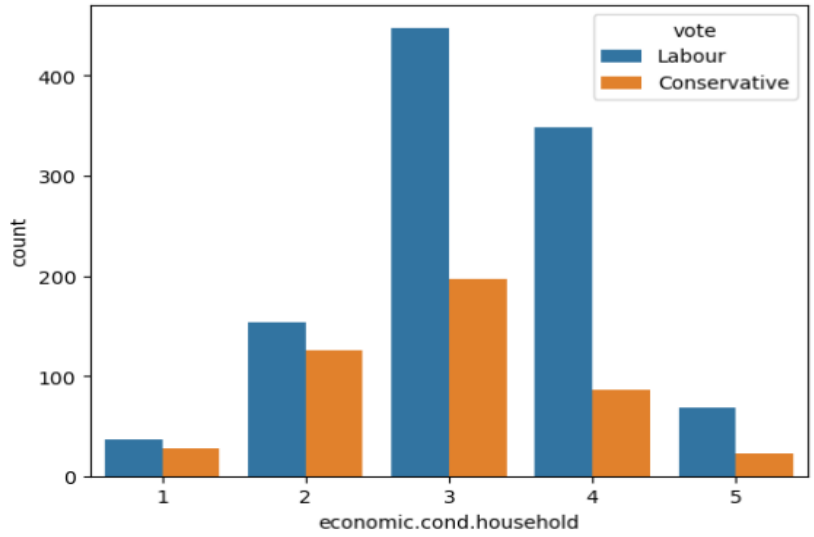


BIVARIATE ANALYSIS :

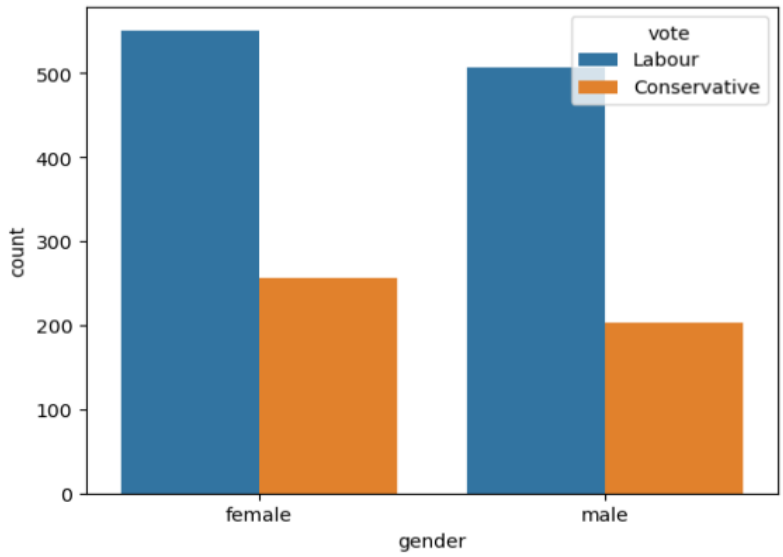
```
<Axes: xlabel='economic.cond.national', ylabel='count'>
```



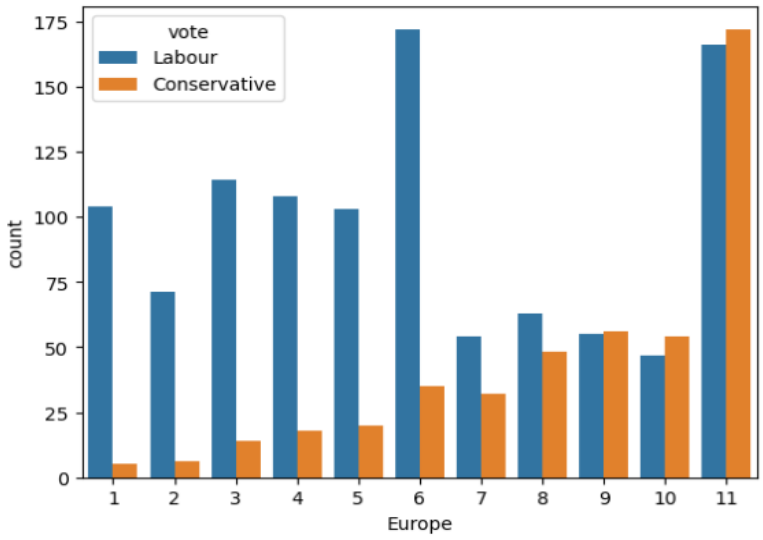
```
<Axes: xlabel='economic.cond.household', ylabel='count'>
```



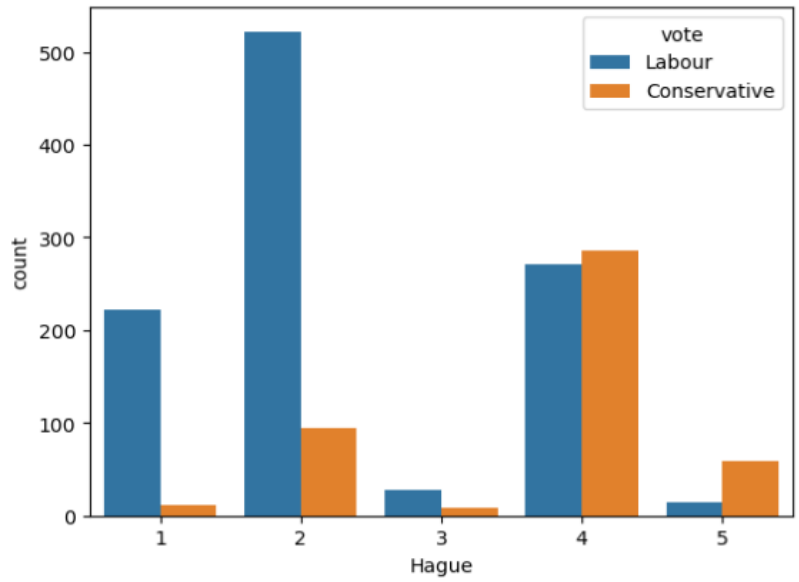

```
<Axes: xlabel='gender', ylabel='count'>
```



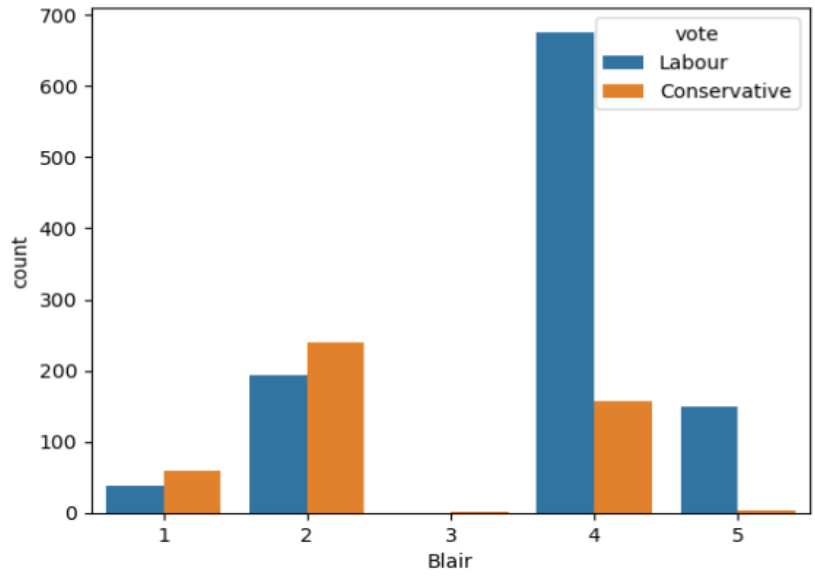

```
<Axes: xlabel='Europe', ylabel='count'>
```



```
<Axes: xlabel='Hague', ylabel='count'>
```



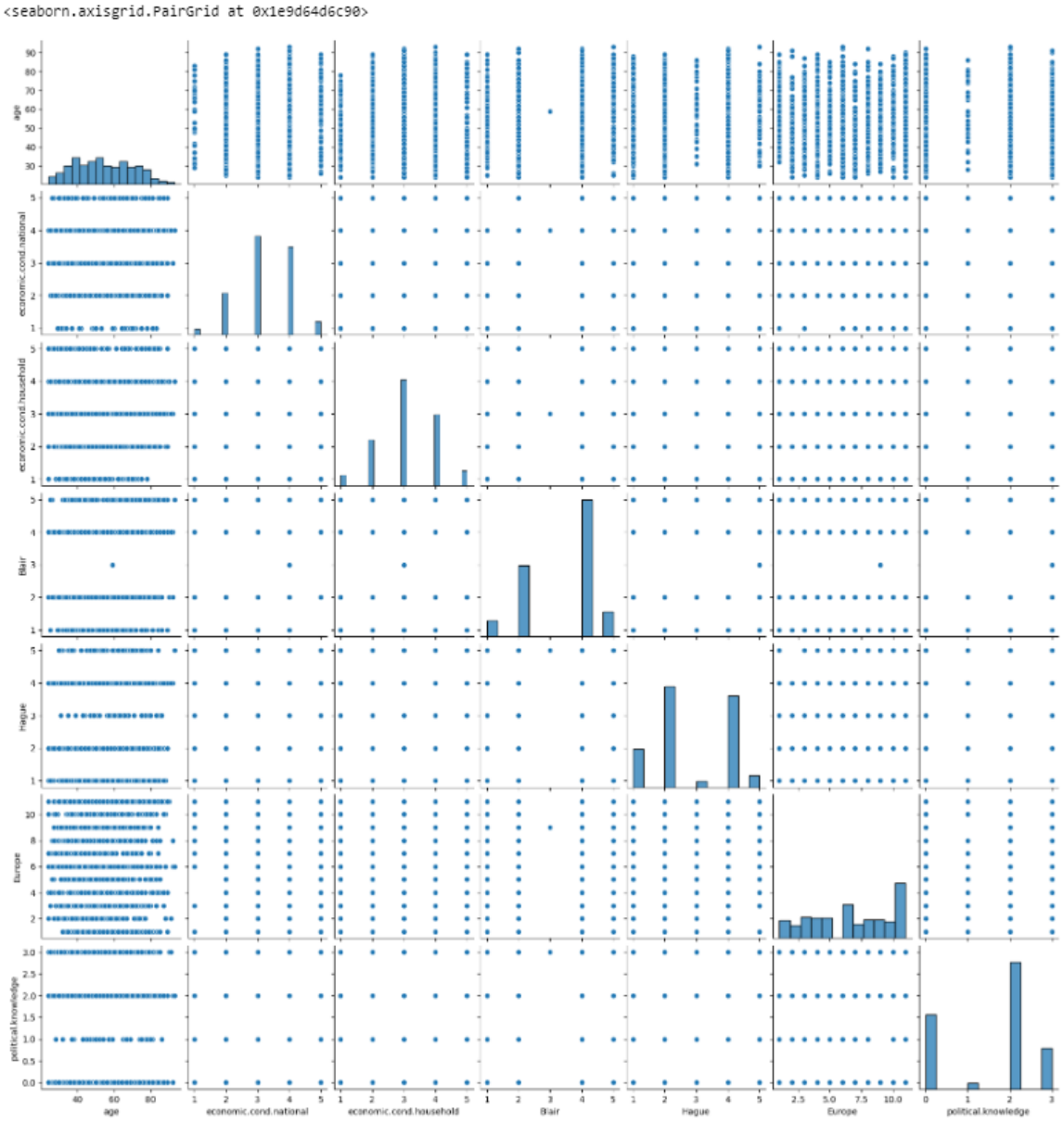

```
<Axes: xlabel='Blair', ylabel='count'>
```



From the above analysis, we get

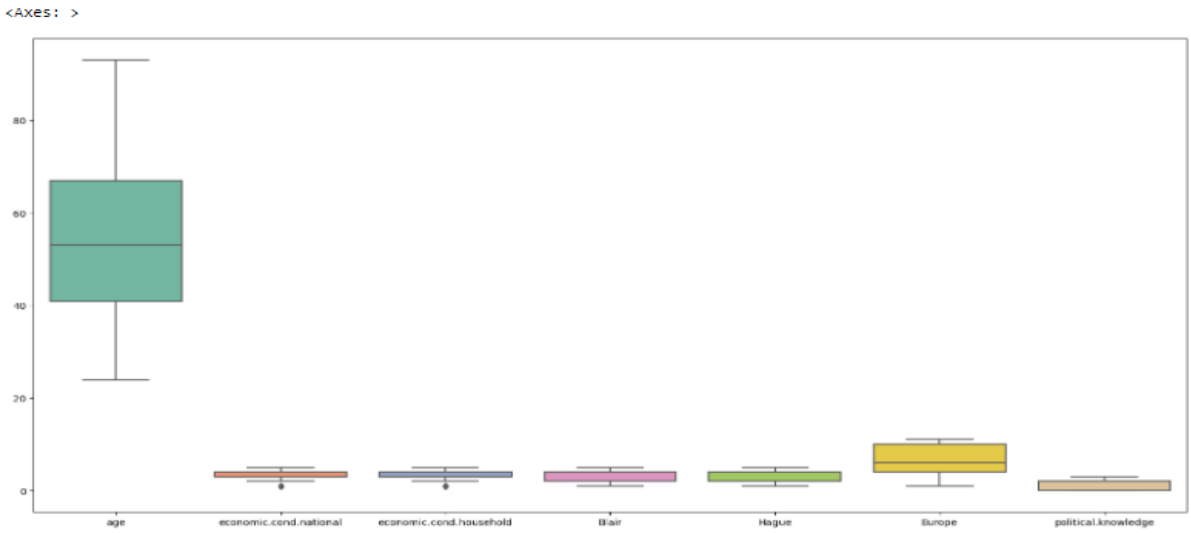
- Labour gets the highest voting from both female and male voters.
- Conservative gets a little bit high votes from Europe ‘11’.
- Almost in all categories labour is getting the maxinum votes.

Pair plot of the data set looks like :



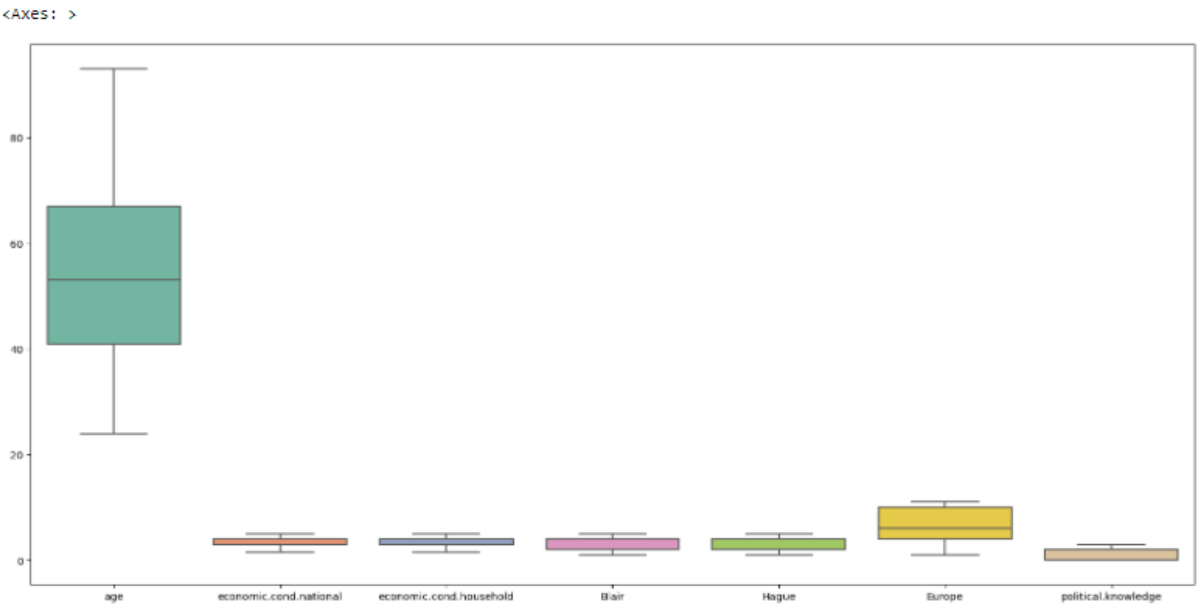
- Pair plot shows the relationship between the variables in the form of scatterplot and the distribution of the variable in the form of histogram .
- In most plots, we will be able to see there is no correlation at all.

Now we create a dataframe that contains only the integer and float type variables and try printing the boxplots for these features.

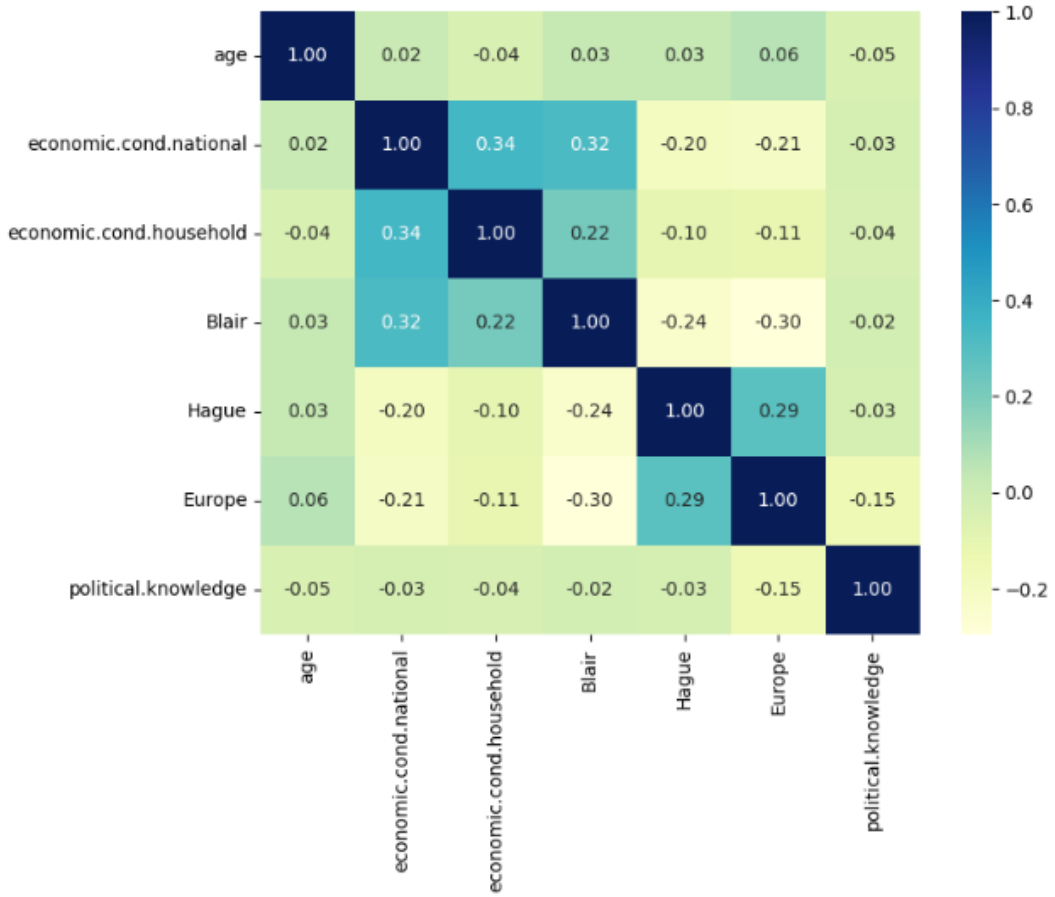


- There are outliers present in the data, these needs to be treated
- There are many methods in which outliers can be treated
- We choose IQR method to treat them
- So, we treat them using IQR method. In this method, any observation that is less than Q1-1.5 IQR or more than Q3 + 1.5 IQR is considered an outlier.

After outlier treatment :



Correlation between variables :



Encoding the data :

The variables ‘vote’ and ‘gender’ are in string, we encode these features to convert it into numerical form fro the modelling purpose.

The first five rows of the dataset after encoding the data:

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	43	3.0	3.0	4	1	2	2	0
1	1	36	4.0	4.0	4	4	5	2	1
2	1	35	4.0	4.0	5	2	3	2	1
3	1	24	4.0	2.0	2	1	4	0	0
4	1	41	2.0	2.0	1	1	6	2	1

The value counts of the feature ‘gender’ :

```
gender
0    808
1    709
Name: count, dtype: int64
```

The value counts of the feature ‘age’ :

```
vote
1    1057
0     460
Name: count, dtype: int64
```

Splitting the data into training and testing data :

- Let us create the x and y variable data with respect to ‘vote’ column as the target variable. Now x having every data except the target variable and y having only the targetvariable .
- We split the independent variables X into two parts, one for training X_train and one for testing X_test
- we split the dependent variable Y into two parts, one for training Y_train and one for the testing Y_test
- Using stats model api as SM to intercept the X variable.
- Using sklearn to split the data into x_train and y_train

LOGISTIC REGRESSION :

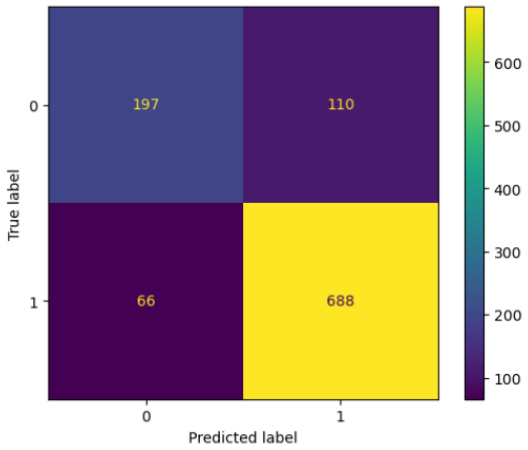
Applyting the logistic regression and fitting the training data

The model score, confusion matrix and classification report of the training data respectively on applying logistic regression is as follows :

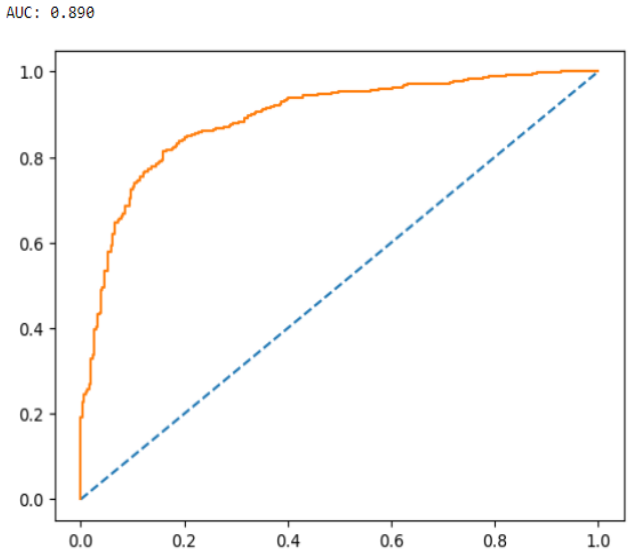
```
0.8341187558906692
[[197 110]
 [ 66 688]]
              precision    recall  f1-score   support

           0       0.75      0.64      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```



The AUC and ROC curve for the training is as follows :

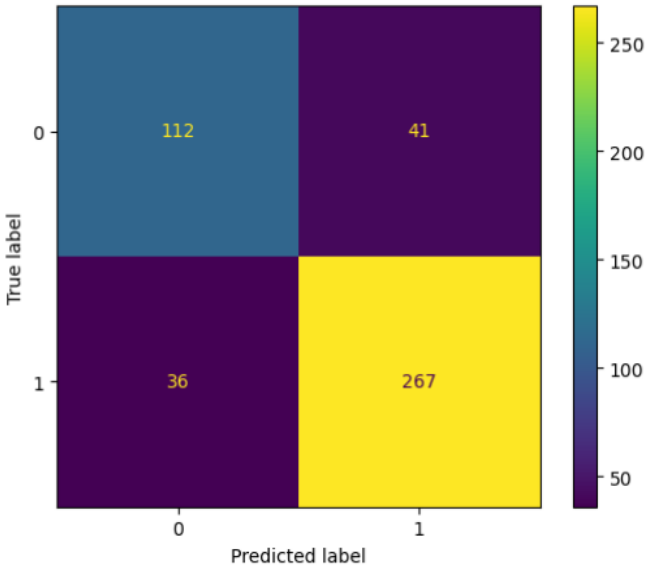


Logistic regression on the test data :

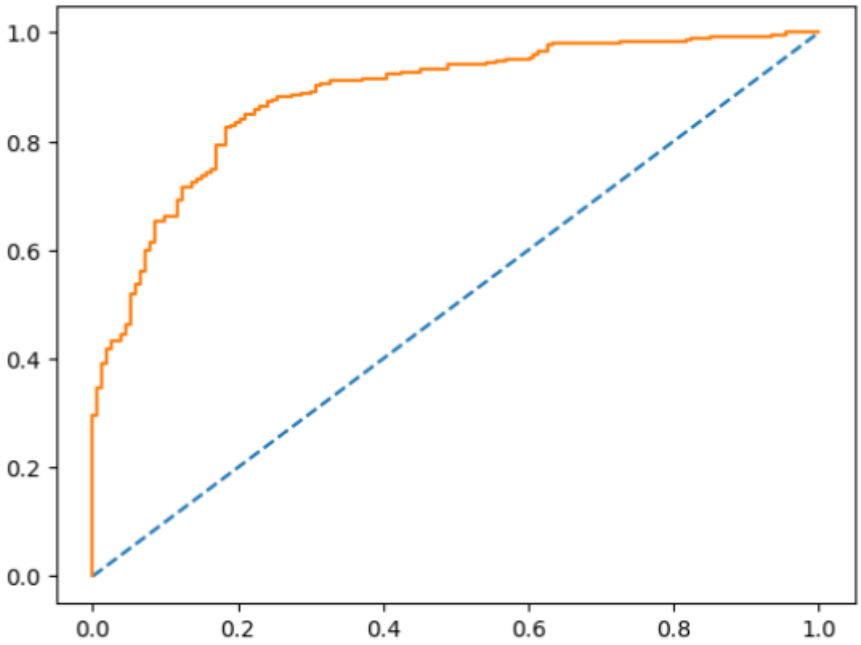
```
0.831140350877193
[[112  41]
 [ 36 267]]
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.87      0.88      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.81      0.81       456
weighted avg       0.83      0.83      0.83       456
```



AUC: 0.890



The model is not overfitting or underfitting. Training and Testing results shows that themodel is excellent with good precision and recall values.

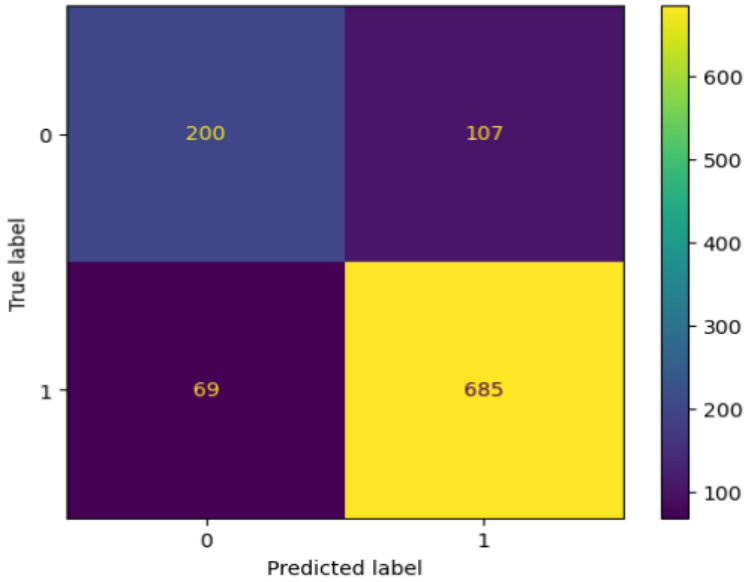
LINEAR DISCRIMINANT ANALYSIS :

Applying linear discriminant analysis on the training data, we get the model score, confusion matrix and classification report as follows:

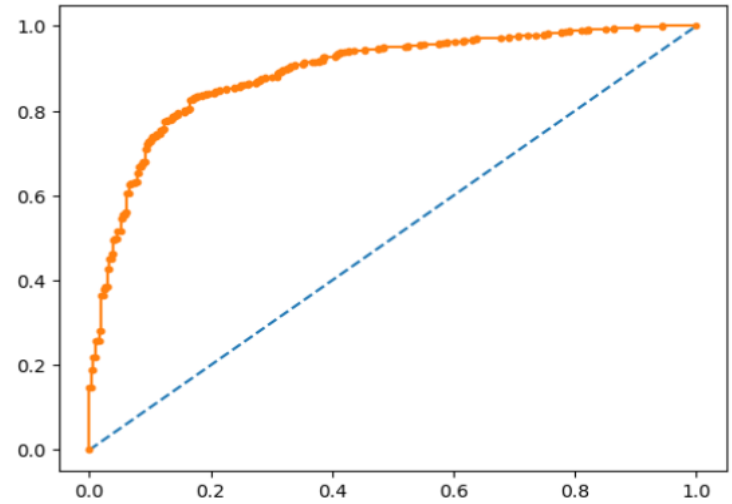
```
0.8341187558906692
[[200 107]
 [ 69 685]]
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```




```
AUC: 0.890
[<matplotlib.lines.Line2D at 0x1e9dbeaa250>]
```

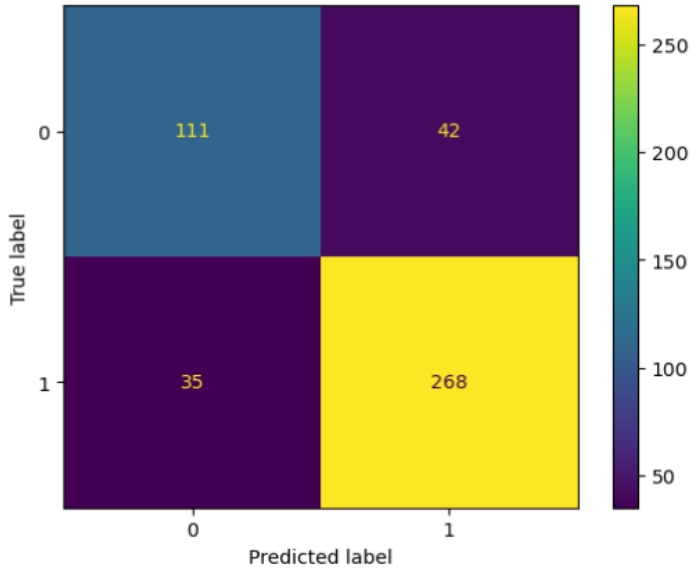


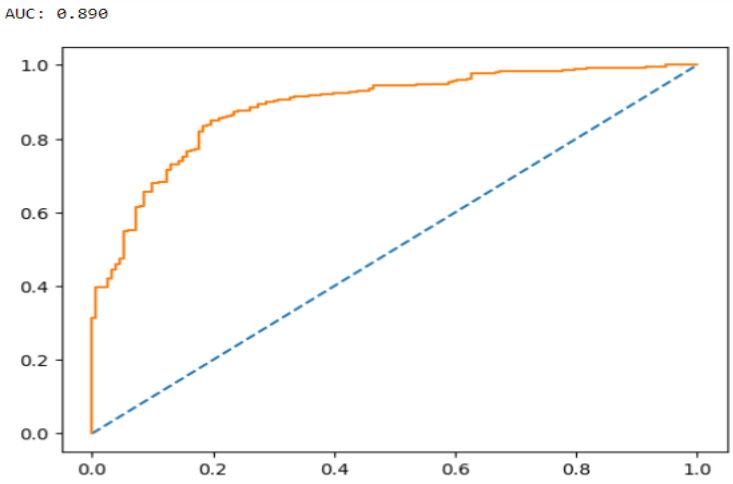
Llinear discriminant analysis into test data :

```
0.831140350877193
[[111  42]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.86      0.88      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.80      0.81       456
weighted avg       0.83      0.83      0.83       456
```





Training and Testing results shows that the model is excellent with good precision and recall values.The LDA model and Logistic regression have almost same performance.

K NEAREST NEIGHBOUR CLASSIFIER

- Scaling the dataset as it is required because KNN is a distance-based algorithm.
- We apply scaling onto the data since KNN involves distance calculation, so we scale the data

The first five rows of the data after scaling :

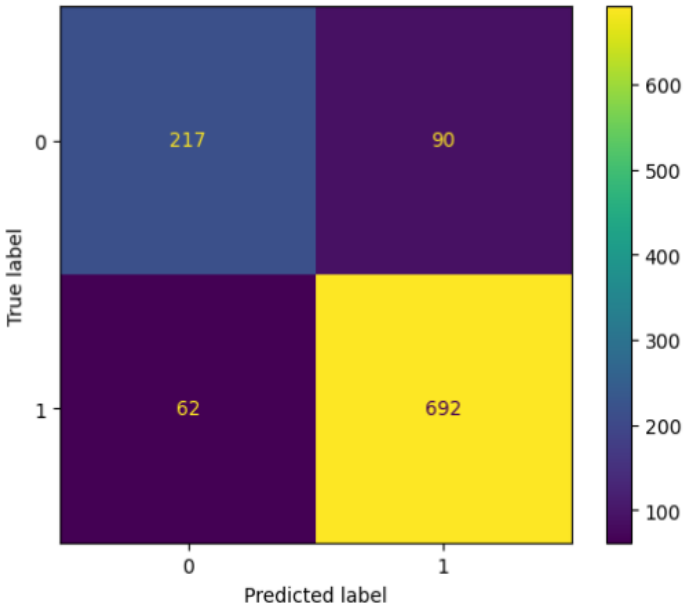
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	-0.716161	-0.301648	-0.179682	0.565802	-1.419969	-1.437338	0.423832	-0.936736
1	-1.162118	0.870183	0.949003	0.565802	1.014951	-0.527684	0.423832	1.067536
2	-1.225827	0.870183	0.949003	1.417312	-0.608329	-1.134120	0.423832	1.067536
3	-1.926617	0.870183	-1.308366	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
4	-0.843577	-1.473479	-1.308366	-1.988727	-1.419969	-0.224465	0.423832	1.067536

Applying KNN modelling into the training data, we get the following model score , confusion matrix, and classification report

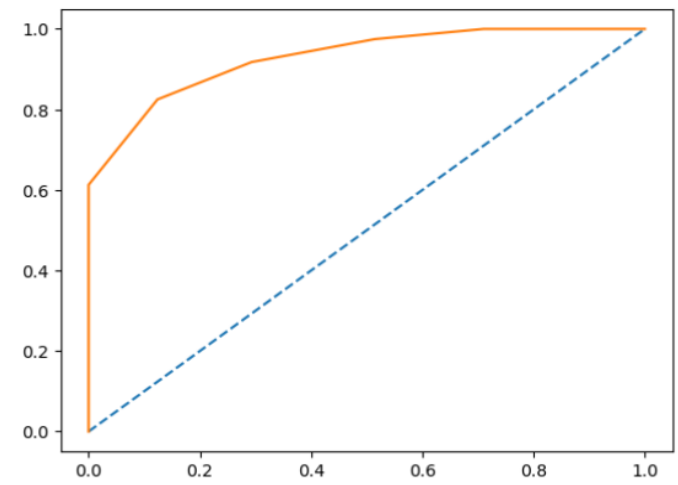
```
0.8567389255419415
[[217  90]
 [ 62 692]]
              precision    recall  f1-score   support

           0       0.78      0.71      0.74       307
           1       0.88      0.92      0.90       754

    accuracy                           0.86      1061
   macro avg       0.83      0.81      0.82      1061
weighted avg       0.85      0.86      0.85      1061
```



AUC: 0.929

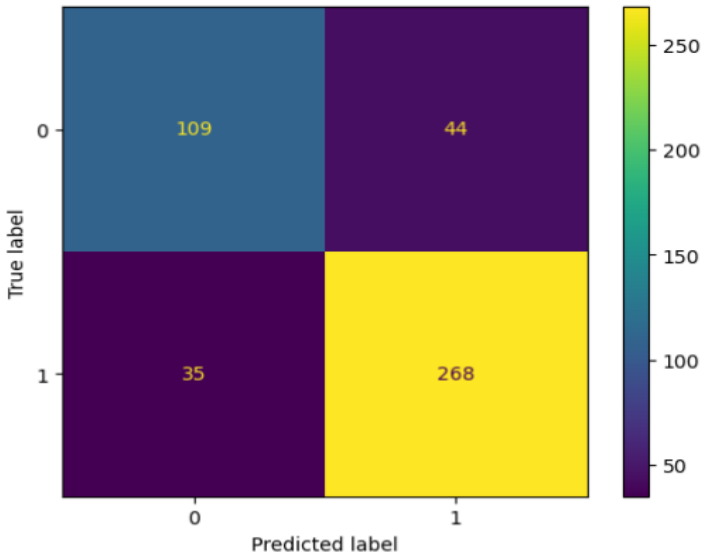


KNN applied onto test data :

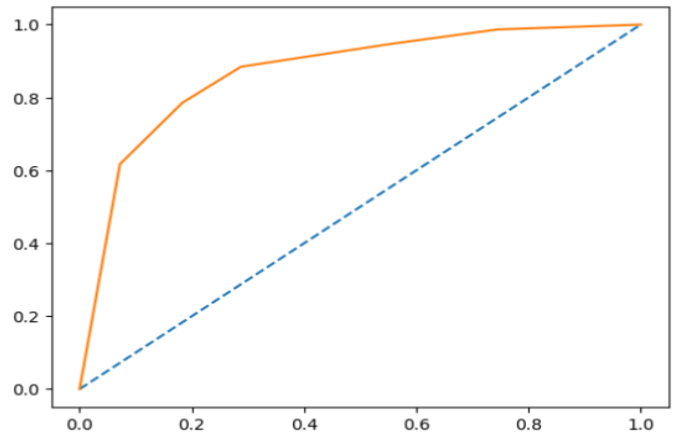
0.8267543859649122

```
[[109  44]
 [ 35 268]]
```

	precision	recall	f1-score	support
0	0.76	0.71	0.73	153
1	0.86	0.88	0.87	303
accuracy			0.83	456
macro avg	0.81	0.80	0.80	456
weighted avg	0.82	0.83	0.83	456



AUC: 0.929



- Training and testing results shows that the model is excellent with good precision and recallvalues.
- This KNN model have good accuracy and recall values.

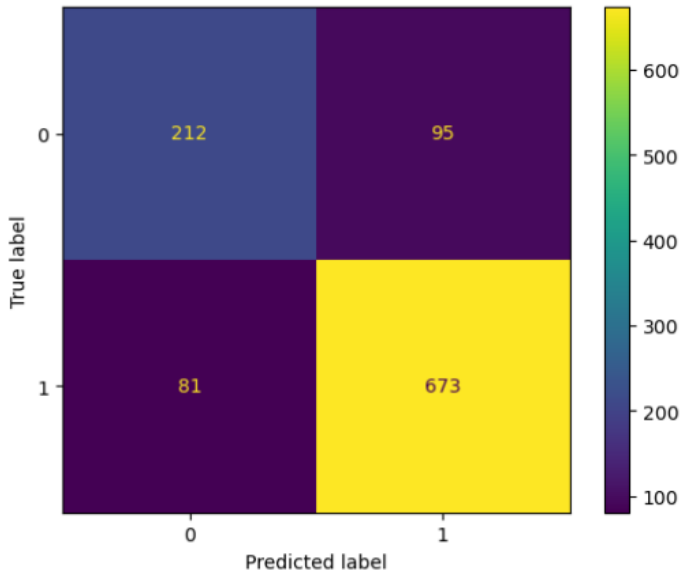
NAÏVE BAYES

- Importing GaussianNB from sklearn and applying NB model.
- Fitting the training data.
- Applying naïve bayes model onto the training data and the model score, confusion matrix and classification report is as follows :

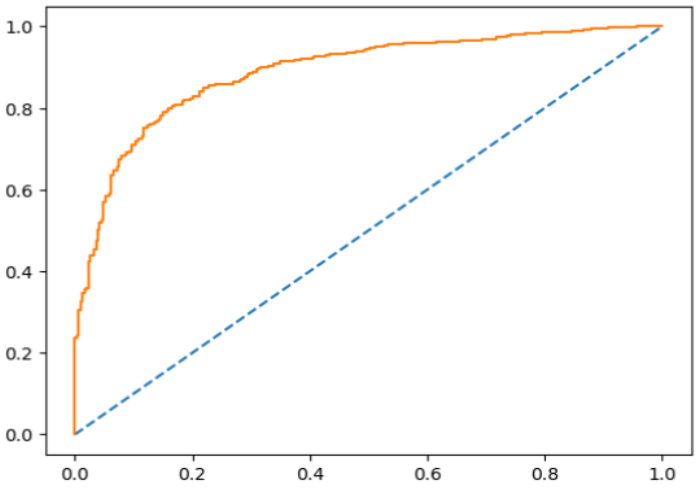
```
0.8341187558906692
[[212  95]
 [ 81 673]]
              precision    recall  f1-score   support

           0       0.72      0.69      0.71       307
           1       0.88      0.89      0.88       754

    accuracy                           0.83      1061
   macro avg       0.80      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```



AUC: 0.889

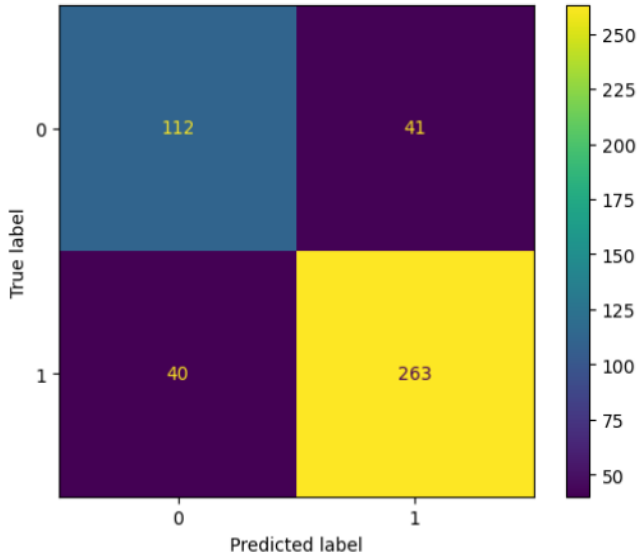


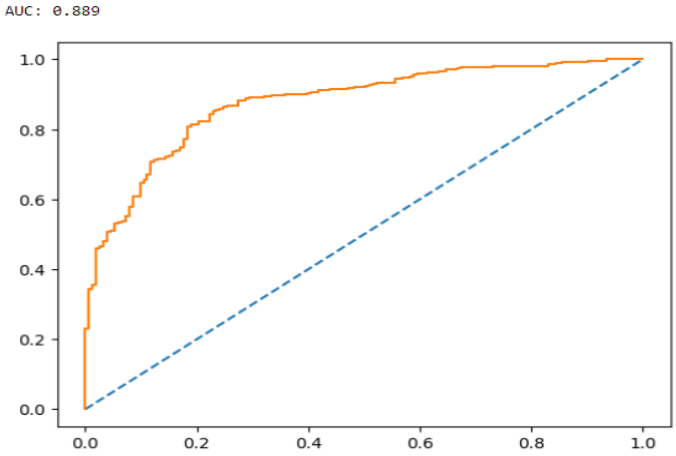
Applying the model onto the test data :

```
0.8223684210526315
[[112  41]
 [ 40 263]]
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.80      0.80       456
weighted avg       0.82      0.82      0.82       456
```





- Training and Testing results shows that the model neither overfitting nor underfitting.The Naive Bayes model also performs well with better accuracy and recall values.
- Even though NB and KNN have same Train and Test accuracy. Based on their recall value in test dataset it is evident that KNN performs better than Naive Bayes.

BAGGING :

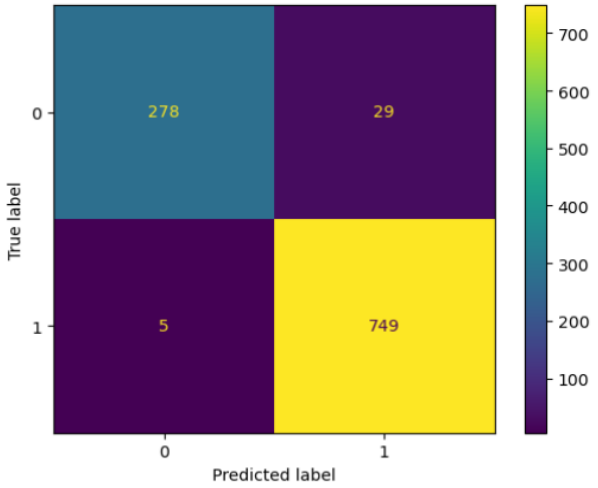
Bagging is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

Bagging reduces variance and helps to avoid overfitting. Using Random forest classifier for bagging below :

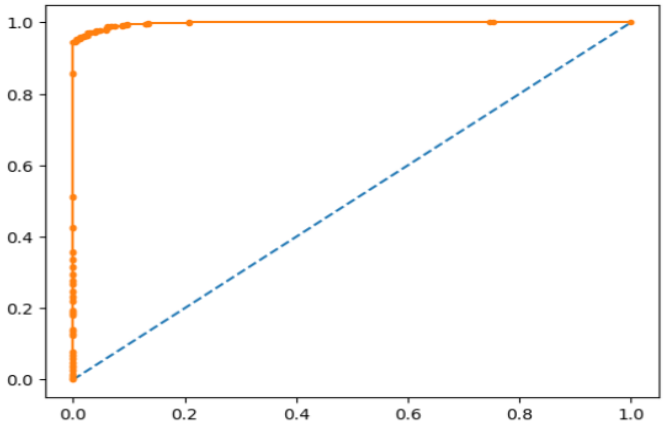
```
0.9679547596606974
[[278  29]
 [  5 749]]
              precision    recall  f1-score   support

           0       0.98      0.91      0.94       307
           1       0.96      0.99      0.98       754

    accuracy                           0.97      1061
   macro avg       0.97      0.95      0.96      1061
weighted avg       0.97      0.97      0.97      1061
```




```
AUC: 0.997
[<matplotlib.lines.Line2D at 0x1e9ecbc07d0>]
```

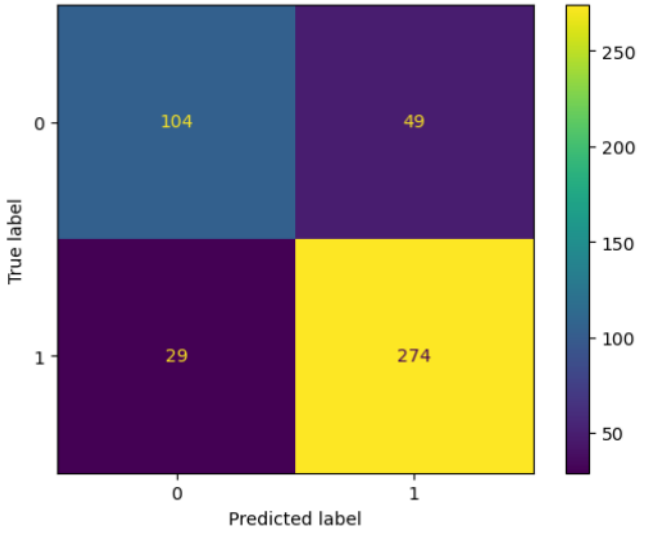


Applying bagging onto the test data:

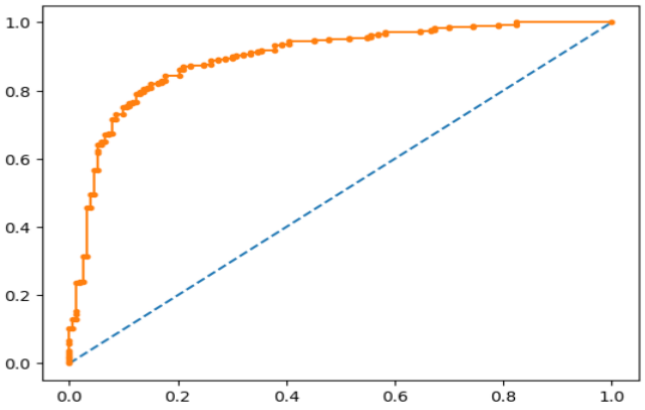
```
0.8289473684210527
[[104  49]
 [ 29 274]]
              precision    recall  f1-score   support

           0       0.78      0.68      0.73       153
           1       0.85      0.90      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```



AUC: 0.896



BOOSTING :

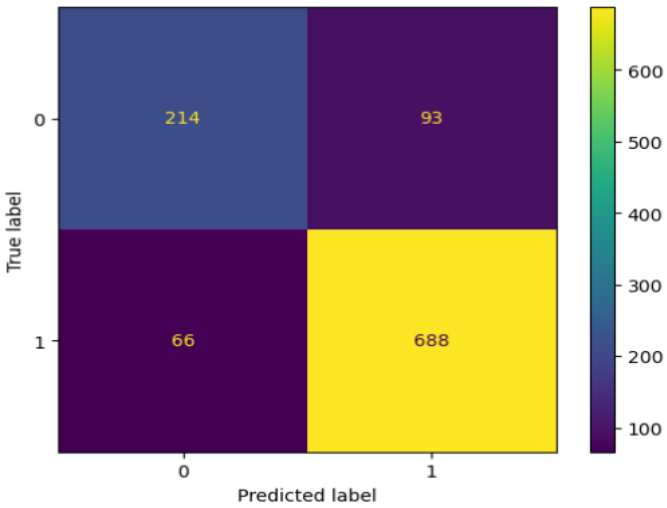
Boosting is an ensemble strategy that consecutively builds on weak learners in order to generate one final strong learner. A weak learner is a model that may not be very accurate or may not take many predictors into account. By building a weak model, making conclusions about the various feature importance and parameters, and then using those conclusionsto build a new, stronger model, Boosting can effectively convert weak learners into a strong learner. The method of boosting used here is Ada boosting (Adaptive boosting).

ADA BOOSTING :

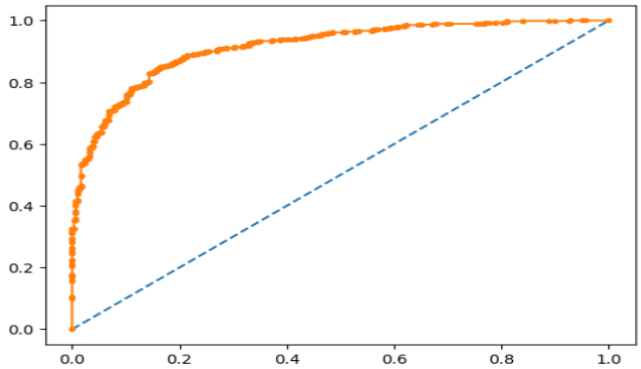
```
0.8501413760603205
[[214  93]
 [ 66 688]]
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       307
           1       0.88      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.85      0.85      0.85      1061
```




```
AUC: 0.915
[<matplotlib.lines.Line2D at 0x1e9ecd9fa90>]
```

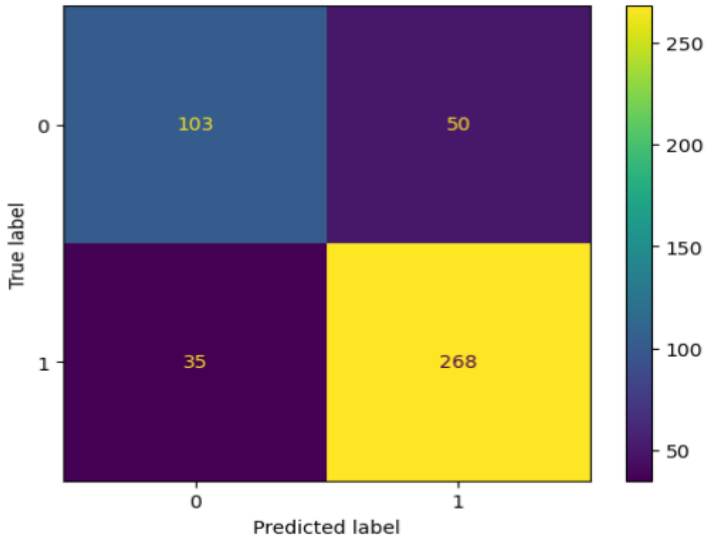


Applying ada boosting onto the test data :

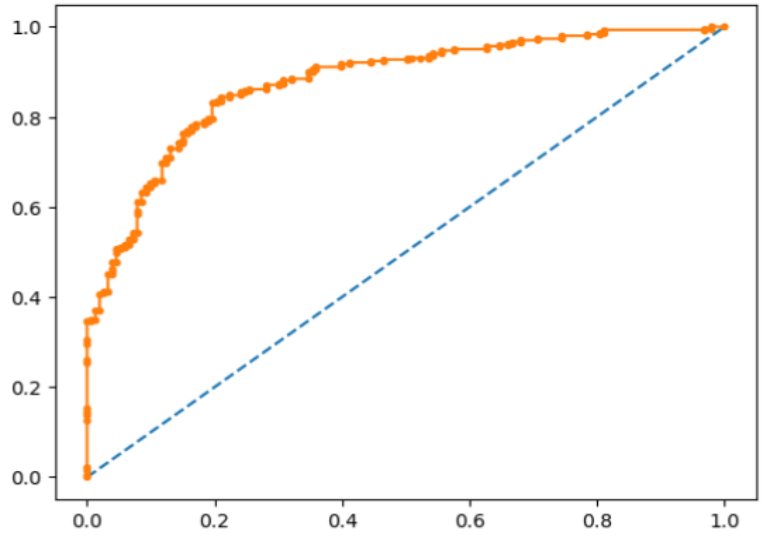
```
0.8135964912280702
[[103  50]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.75      0.67      0.71       153
           1       0.84      0.88      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.78      0.79       456
weighted avg       0.81      0.81      0.81       456
```



AUC: 0.877

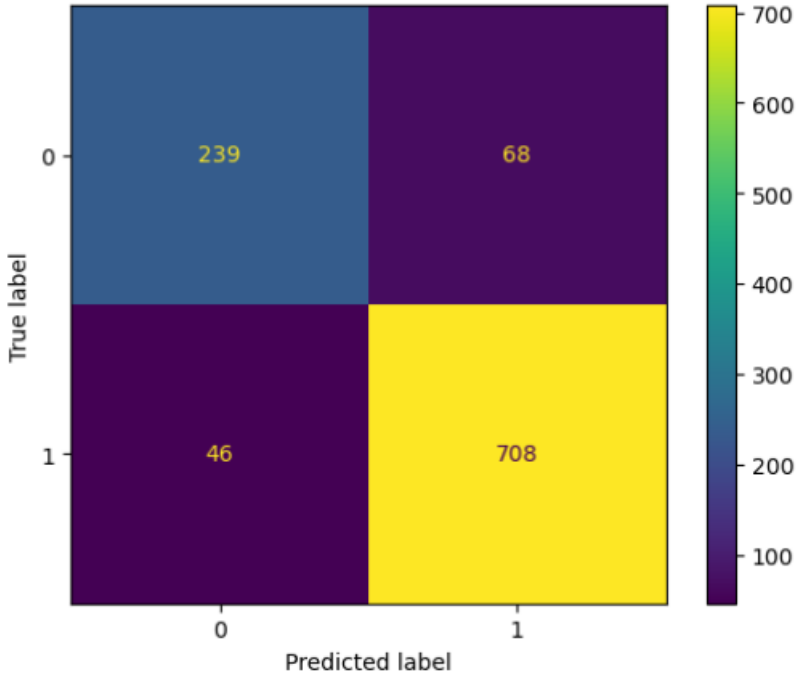


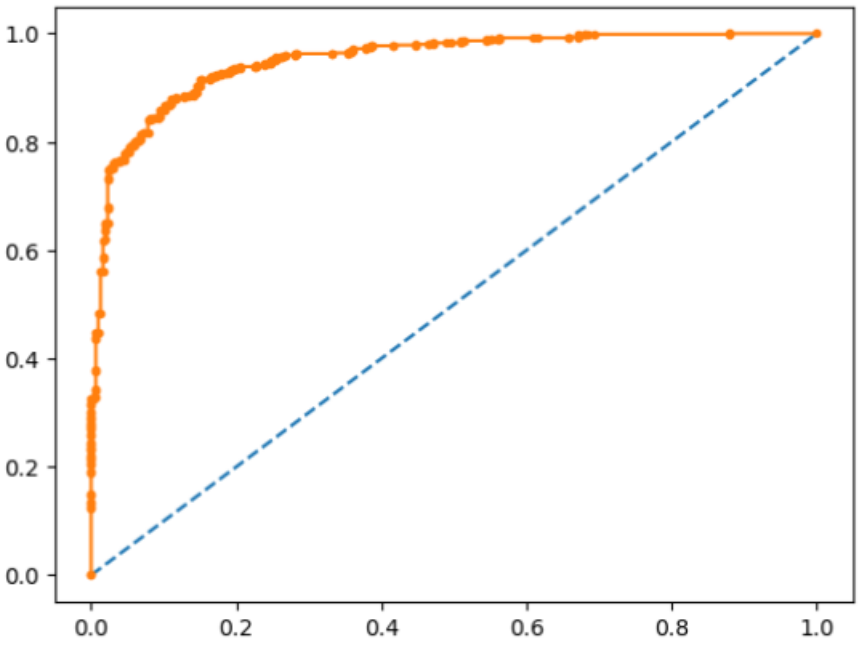
GRADIENT BOOSTING:

```
0.8925541941564562
[[239  68]
 [ 46 708]]
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.87      1061
weighted avg       0.89      0.89      0.89      1061
```



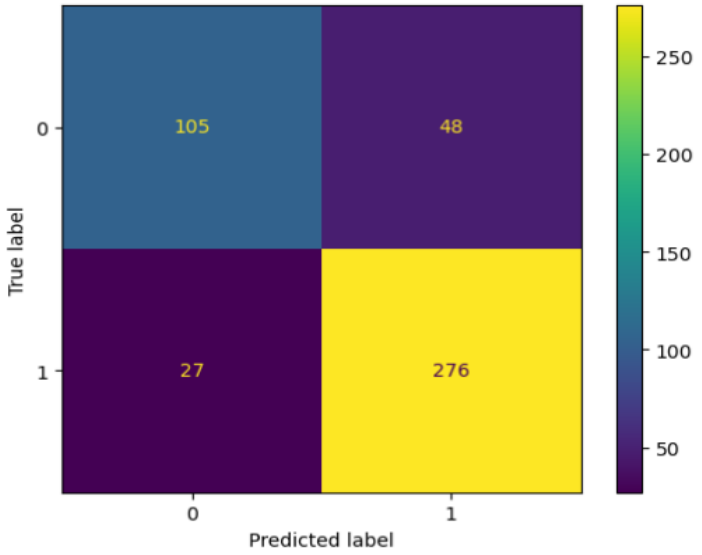


Applying gradient boosting onto the test data:

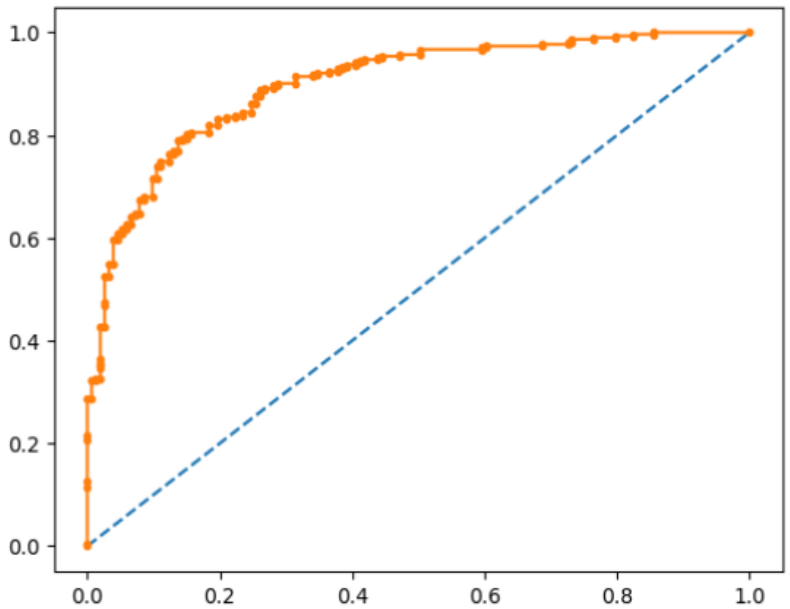
```
0.8355263157894737
[[105  48]
 [ 27 276]]
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       153
           1       0.85      0.91      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.84      0.83       456
```




```
AUC: 0.899
```



Comparison of Models

Let's look at the performance of all the models on the Train Data set

Recall refers to the percentage of total relevant results correctly classified by the algorithm and hence we will compare Recall of class "1" for all models.

Naive Bayes - Recall for class "1" is .89

```
0.8341187558906692
[[212  95]
 [ 81 673]]
              precision    recall  f1-score   support

           0       0.72      0.69      0.71       307
           1       0.88      0.89      0.88       754

    accuracy                           0.83      1061
   macro avg       0.80      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

KNN - Recall for class "1" is .92

```
0.8567389255419415
[[217  90]
 [ 62 692]]
              precision    recall  f1-score   support

           0       0.78      0.71      0.74       307
           1       0.88      0.92      0.90       754

    accuracy                           0.86      1061
   macro avg       0.83      0.81      0.82      1061
weighted avg       0.85      0.86      0.85      1061
```

Bagging - Recall for class "1" is .99

```
0.9679547596606974
[[278  29]
 [  5 749]]
              precision    recall  f1-score   support

           0       0.98      0.91      0.94       307
           1       0.96      0.99      0.98       754

    accuracy                           0.97      1061
   macro avg       0.97      0.95      0.96      1061
weighted avg       0.97      0.97      0.97      1061
```

Gradient Boosting - Recall for class "1" is .94

```
0.8925541941564562
[[239  68]
 [ 46 708]]
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.87      1061
weighted avg       0.89      0.89      0.89      1061
```

so as per the train data,

- low performing models are – Naïve bayes
- best performing models are -Boosting and bagging

Let's look at the performance of all the models on the Test Data set

Recall on the Test Data Set

Naive Bayes - Recall for class "1" is .87

```
0.8223684210526315
[[112  41]
 [ 40 263]]
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.80      0.80       456
weighted avg       0.82      0.82      0.82       456
```

KNN - Recall for class "1" is .88

```
0.8267543859649122
[[109  44]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       153
           1       0.86      0.88      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.80      0.80       456
weighted avg       0.82      0.83      0.83       456
```

Bagging - Recall for class "1" is .90

```
0.8289473684210527
[[104  49]
 [ 29 274]]
              precision    recall  f1-score   support

           0       0.78      0.68      0.73       153
           1       0.85      0.90      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

Gradient Boosting - Recall for class "1" is .91

```
0.8355263157894737
[[105  48]
 [ 27 276]]
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       153
           1       0.85      0.91      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.84      0.83       456
```

so as per the test data,

- low performing models are – Naïve bayes
- best performing models are – Gradient boosting

Insights based on the predictions :

- Based on the predictions from the given data, we can say that Labour party will be winning the upcoming elections with more no of seats compared to Conservative party.
- If we look at the model, we can see Labour party has high number of rating in terms of economic house conditions and national economic conditions. If the opposition party i.e., Conservative party can offer much better economic conditions and make the people believe in their abilities, they can increase theno of votes, and gain much more seats.
- Also, from the model, we can see that each voters are having very less knowledge of politics. Therefore campaings should be conducted so that the people know more about the political party they are choosing.
- After the elections, political leaders should make sure that they stand by their words, so that during the next elections too they can win without much competition.
- People should have a choice to choose their own leaders. If we look at the rating given for the leaders,we can see conservative party is having very less rating. Therefore the party should make it a point to either change the leader or to bring in more changes in the party.

PROBLEM 2

Define the problem and Perform Exploratory Data Analysis

Problem Definition - Find the number of Character, words & sentences in all three speeches

- There are 7571 characters and 1360 words in Rossevelt file
- There are 7618 characters and 1390 words in Kennedy file
- There are 9991 characters and 1819 words in Nixon file

- The number of sentences in the Rossevelt file is 67
- The number of sentences in the Kennedy file is 52
- The number of sentences in the Nixon file is 68

Text cleaning

Stopword removal - Stemming - find the 3 most common words used in all three speeches

- Now we remove all of the stop words present in all the speeches and we print only the data with no stop words in all the speeches.
- Now for stemming we use porter stemmer to stem the words present in the speech
- Now we use RegexpTokenizer from nltk to find out the most commom words used in all the three speeches

  The 3 most common words and their frequency used in Rossevelt's speech are :

  ```
  [('Nation', 12), ('Spirit', 9), ('Life', 9)]
  ```

  The 3 most common words and their frequency used in Kennedy's speech are :

  ```
  [('World', 8), ('Sides', 8), ('Pledge', 7)]
  ```

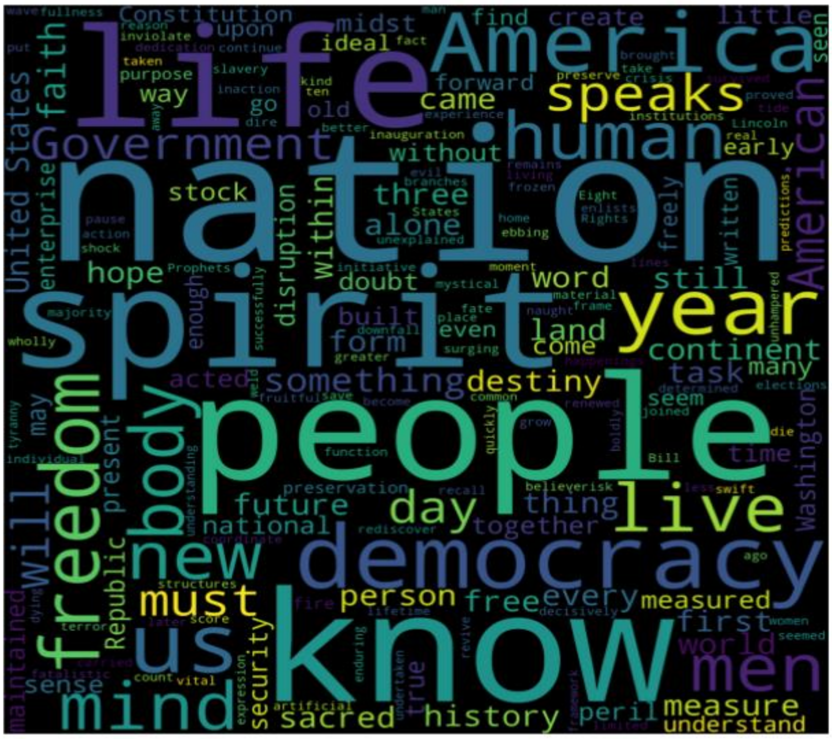
  The 3 most common words and their frequency used in Nixon's speech are :

  ```
  [('America', 21), ('Peace', 19), ('World', 18)]
  ```

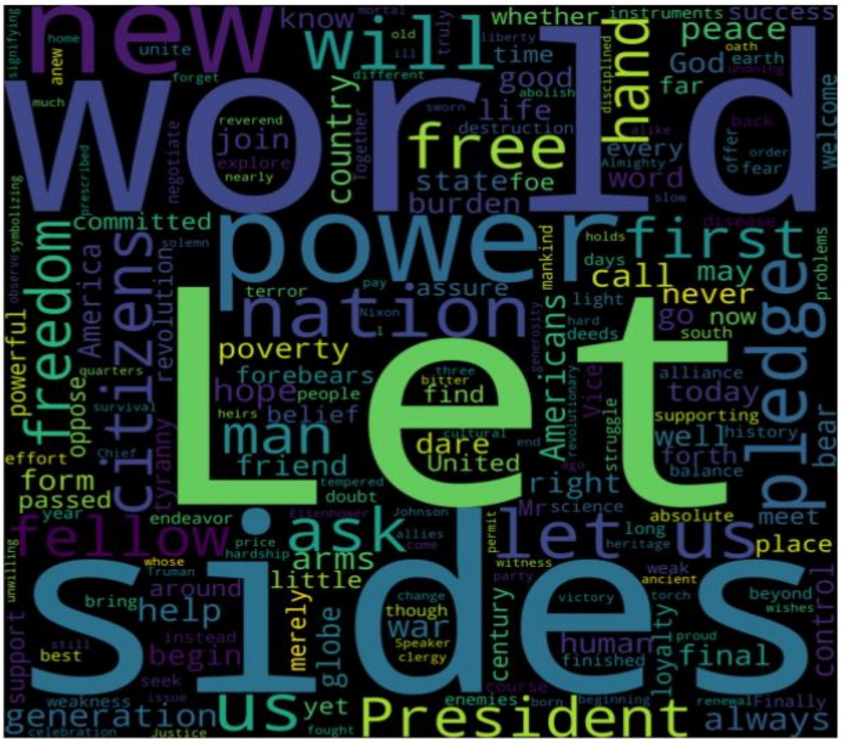
Plot Word cloud of all three speeches

Show the most common words used in all three speeches in the form of word clouds

The word cloud for Rossevelt :



The word cloud for Kennedy :



The word cloud for Nixon :